

Spike Clustering and Neuron Tracking over Successive Time Windows

Michael T. Wolf

Mechanical Engineering
California Institute of Technology
wolf@caltech.edu

Joel W. Burdick

Mechanical Engineering and Bioengineering
California Institute of Technology
jwb@robotics.caltech.edu

Abstract—This paper introduces a new methodology for tracking signals from individual neurons over time in multi-unit extracellular recordings. The core of our strategy relies upon an extension of a traditional mixture model approach, with parameter optimization via expectation-maximization (EM), to incorporate clustering results from the preceding time period in a Bayesian manner. EM initialization is also achieved by utilizing these prior clustering results. After clustering, we match the current and prior clusters to track persisting neurons. Applications of this spike sorting method to recordings from macaque parietal cortex show that it provides significantly more consistent clustering and tracking results.

I. INTRODUCTION

The need to reliably identify and track the activities of a single neuron in multi-unit recordings is a common problem in basic electrophysiological studies and engineered neural interfaces. In chronic multi-electrode implants that are used as the front end for neural prostheses, mechanical shocks can cause small repositionings of the implanted electrodes, leading to appreciable changes in the amplitude, phase, and numbers of neural signals that are recorded by each electrode [1]. These changes complicate the process of reliably identifying and monitoring individual neurons, whose previously calibrated characteristics are the basis for neural decoding algorithms that drive the neural prosthesis. In related work, the authors have developed algorithms and robotic microdrive devices to autonomously position electrodes so as to initially isolate and then maintain high quality neural recordings for both acute and chronic electrode recordings [2], [3], [4], [5]. As the autonomous electrode continually repositions itself to maintain a high quality recording signal, a similar problem arises: previously identified neurons must be re-identified in the current recording interval, despite changes in their waveforms due to electrode movement (and other reasons).

In general, the interpretation of all extracellular recordings requires a process to associate the experimental data with the activity of a particular neuron over the duration of the recording, commonly referred to as “spike sorting” (see [6] for a review of the challenge). A signal, S , of length T is sampled from an electrode. After spikes are detected in S and temporally aligned, the spike waveforms are often projected onto an d -dimensional feature space (usually a 2-dimensional principal component (PCA) basis), where each waveform is represented as a point. These points are “clustered” into sets, each assumed to be associated with a unique neuron in the multi-unit signal, using any of several techniques,

including hierarchical [7], k-means [8], neural networks [9], and statistical mixture models [10], [11]. As alternatives to PCA features, some authors have favored wavelet-based representations [12], [13], [14], [15], phase-space techniques [16], or Bayesian approaches on the full waveform [17]. Because of our motivating interest in autonomous microdrives and chronic neural interfaces, we require *unsupervised* methods applicable to *online* recording — or at least to small, real-time batches. That is, at time $t_0 + T$, an additional electrode signal sample of duration T is taken and processed in similar fashion; our goal is to reliably track individual neurons across successive sampling intervals (time windows) at t_0 , $t_0 + T$, $t_0 + 2T$, and so on.

The non-stationarity of spike waveforms, primarily due to electrode drift, is a commonly cited culprit for neuron tracking difficulties (or “clustering difficulties” for long T) [10], [18], [19]. However, when the recording application involves repeated sampling over time (or when a long T is split into many intervals of length Δt for analysis), our experience has shown that the inconsistency of conventional clustering method’s output is a crucial issue, as each sampling step’s clusters must be matched to those in the preceding and subsequent step(s). Essentially all conventional spike sorting methods based on clustering fail to integrate the available information over time to increase spike clustering consistency. That is, the results of clustering one set of sampled data at t_k is not used to improve the clustering of the subsequent set of sampled data at $t_k + T$, etc. (Bar-Hillel et al. [10] includes neighboring clustering results but uses *future* as well as past steps in a batch process.) More generally, due to the many challenges of unsupervised clustering, data sampled from consecutive time windows that may be deemed similar by humans will often be clustered much differently by computers, and the over- and under-clustering of spikes often changes each cluster’s statistics enough to significantly reduce the reliability of matching the clusters in consecutive time windows. These inconsistencies limit the effectiveness of automated probes and neural interfaces.

In this paper we mitigate this effect by incorporating prior information into a Bayesian clustering algorithm. Our strategy’s foundation is the optimization of the model parameters of a Gaussian mixture via expectation-maximization (EM) to best represent the probability distributions of the signal-generating neurons [20], [21], [22]. Under the assumption that the preceding time window has a reasonable clustering

result, we first use the preceding model's statistics to guess the initial values (or seed) for the EM algorithm. Then we incorporate the preceding cluster locations as a prior during the EM algorithm, thus seeking maximum-*a-posteriori* (MAP) rather than maximum likelihood (ML) results. Importantly, we implement the prior in a manner such that the method will likely succeed even if the preceding clustering was incorrect or if different neurons' signals are recorded during the two intervals. Finally, we match the current and prior clusters to track persisting neurons. Note this procedure assumes the clusters are approximately stationary over small time intervals, with cluster centers perhaps executing a modest random walk between time steps. We have applied this method to several recordings from macaque parietal cortex, and our results show that it gives much more consistent clustering.

II. MIXTURE MODEL OPTIMIZATION VIA EM

While many traditional clustering procedures have been adapted to classify neural waveforms, the optimization of a (typically Gaussian) mixture model [20] has been shown to be an effective, and often superior, approach [6], [10], [11]. An attractive underlying assumption is that each cluster's spikes can be modeled as samples from a different multivariate statistical distribution, where each distribution represents the signal features of a generating neuron.

Let us first review the classical mixture model and corresponding ML optimization. Following the notation of [23], we consider the *mixture likelihood*, \mathcal{L}_M , of the model parameters given the data:

$$\mathcal{L}_M(\theta_1, \dots, \theta_G; \tau_1, \dots, \tau_G \mid X) = \prod_{i=1}^n \sum_{k=1}^G \tau_k f_k(x_i \mid \theta_k), \quad (1)$$

where:

- X is the set of n (spike) observations $x_i \in \mathbb{R}^d$ in a d -dimensional feature space taken during sampling interval T . In this work, we use the first two principal components ($d = 2$).
- G is the number of mixture components (clusters) the EM algorithm seeks (for now considered known *a priori* but discussed further in section III-B).
- τ_k is the probability that any spike observation belongs to component k (i.e. generated by source neuron k), with $\tau_k \geq 0$ and $\sum_{k=1}^G \tau_k = 1$.
- f_k is the probability density of the k^{th} mixture component, herein assumed Gaussian.
- θ_k are the parameters of k^{th} mixture component (mean and covariance matrix for a Gaussian mixture, $\theta_k = \{\mu_k, \Sigma_k\}$).

The powerful EM algorithm [24] is typically applied to estimate the mixture parameters by log-likelihood maximization. At the same time, the EM procedure assigns data points to the appropriate mixture component, thereby effecting the separation of spikes. To apply this technique, we view our data X as "incomplete" and augment it by Z , the set of

membership variables $z_i = (z_{i1}, \dots, z_{iG})$,

$$z_{ik} = \begin{cases} 1 & \text{if } x_i \text{ belongs to cluster } k \\ 0 & \text{otherwise.} \end{cases}$$

Incorporating Z and denoting $\Theta = \{\theta_k, \tau_k\}_{k=1}^G$ we can derive the corresponding *complete-data log-likelihood*

$$l_{CD}(\Theta \mid X, Z) = \sum_{i=1}^n \sum_{k=1}^G z_{ik} \log [\tau_k f_k(x_i \mid \theta_k)]. \quad (2)$$

The EM algorithm iterates between an E-step to calculate the conditional expectation $\hat{z}_{ik} = E[z_{ik} \mid x_i, \Theta] \in [0, 1]$ using the current parameter estimates and an M-step to find the parameter estimates Θ that maximize (2) given \hat{z}_{ik} , until some convergence threshold is reached. The algorithm requires an initial guess or "seed clusters" for initialization, and a seeding method is discussed in detail below.

III. UTILIZING PRIOR INFORMATION

Our primarily technical innovation lies in converting the EM algorithm to MAP optimization (rather than ML) for the purpose of improved cluster tracking throughout the recording session. Although Cheeseman and Stutz [22] previously proposed MAP optimization for generic clustering cases, we explicitly derive the EM adjustments for a mixture prior appropriate to our application. Additionally, our method uses the prior clusters to generate the EM seed clusters, which greatly increases our chances of avoiding poor local optima while still mindful of phenomena commonly encountered in clustering neural data over time. Finally, a simple tracking method is implemented to estimate whether clusters in consecutive time windows, or *steps*, represent the same neuron.

A. Extending EM for Cluster Location Priors

To include prior information into our EM formulation, we naturally begin with Bayes' Rule,

$$p(\Theta \mid X) \propto p(X \mid \Theta)p(\Theta). \quad (3)$$

First, let us construct the appropriate prior. Independence of each cluster is a reasonable assumption, so $p(\Theta) = p(\theta_1, \tau_1) \dots p(\theta_G, \tau_G)$, as is the independence of the individual statistics of each distribution, $p(\theta_k, \tau_k) = p(\mu_k)p(\Sigma_k)p(\tau_k)$. Thus,

$$p(\Theta) = p(\mu_1)p(\Sigma_1)p(\tau_1) \dots p(\mu_G)p(\Sigma_G)p(\tau_G),$$

where $p(\mu_k), p(\Sigma_k), p(\tau_k)$ are the prior probabilities of the respective mixture model parameters.

Priors may be constructed on any parameter, but for tracking purposes the most important one is the location of each cluster center, μ_k . Each Σ_k and τ_k may vary more substantially from one step to the next without much concern. Thus we give uniform priors to all Σ_k and τ_k and they drop out as constants,

$$p(\Theta) \propto p(\mu_1) \dots p(\mu_G).$$

(However, priors on Σ_k and τ_k may be constructed and incorporated into our method in a similar manner.) We look

for the k^{th} mean μ_k to be near to *any* of the preceding step's cluster locations, without regard to which one. So we place the same prior on each μ_k , namely an equitable mixture of Gaussians representing all \tilde{G} of the previous step's means (denoted $\tilde{\mu}_j$):

$$p(\mu_k) = p(\mu_k | \{\tilde{\mu}_j, \Sigma_{\tilde{\mu}_j}\}_{j=1}^{\tilde{G}}) = \sum_{j=1}^{\tilde{G}} \frac{1}{\tilde{G}} f_j(\mu_k | \tilde{\mu}_j, \Sigma_{\tilde{\mu}_j})$$

where f_j is the d -dimensional multivariate normal density, \tilde{G} is the number of clusters identified in the previous step, and $\Sigma_{\tilde{\mu}_j}$ is the covariance associated with our estimation of the prior mean $\tilde{\mu}_j$, as $\tilde{\mu}_j$ is itself a quantity estimated from the samples of the last time step. Note that, if using data-dependent features such as PCA, we clearly must ensure the prior statistics and current data are in the same coordinate frame — we convert the prior step's full-dimension spike data to the current PCA space, then calculate the prior clusters' statistics in this space.

Thus we have the completed prior

$$p(\Theta) \propto \prod_{k=1}^G \sum_{j=1}^{\tilde{G}} \frac{1}{\tilde{G}} f_j(\mu_k | \tilde{\mu}_j, \Sigma_{\tilde{\mu}_j}), \quad (4)$$

which bears distinct resemblance to our mixture likelihood (1) and, when incorporated into (3), will in fact share the same difficulty of maximization. We can, however, apply the same solution: add hidden variables and optimize via EM. Denote the new “membership” variable $y_k = (y_{k1}, \dots, y_{k\tilde{G}})$, essentially indicating whether the prior of previous cluster j should influence the current cluster k , or, ideally,

$$y_{kj} = \begin{cases} 1 & \text{if } \mu_k \text{ and } \tilde{\mu}_j \text{ represent the same neuron} \\ 0 & \text{otherwise.} \end{cases}$$

Thus we can derive the complete-data prior log density on the means:

$$\log p(\mu, Y | \tilde{\Theta}) = \sum_{k=1}^G \sum_{j=1}^{\tilde{G}} y_{kj} \log \left[\frac{1}{\tilde{G}} f_j(\mu_k | \tilde{\mu}_j, \Sigma_{\tilde{\mu}_j}) \right]. \quad (5)$$

Returning to (3) and expanding to include our hidden variables (with $\tilde{\Theta} = \{\tilde{\mu}_j, \Sigma_{\tilde{\mu}_j}\}_{j=1}^{\tilde{G}}$), we have

$$p(\Theta, Y | X, Z, \tilde{\Theta}) \propto p(X, Z | \Theta, Y, \tilde{\Theta}) p(\Theta, Y | \tilde{\Theta}). \quad (6)$$

We will maximize the log posterior and so take the log of (6) and substitute in (2) and (5),

$$\begin{aligned} \log p(\Theta, Y | X, Z, \tilde{\Theta}) &= \log p(X, Z | \Theta, Y, \tilde{\Theta}) + \log p(\Theta, Y | \tilde{\Theta}) + \mathcal{C} \\ &= \sum_{i=1}^n \sum_{k=1}^G z_{ik} \log [\tau_k f_k(x_i | \mu_k, \Sigma_k)] \\ &\quad + \sum_{k=1}^G \sum_{j=1}^{\tilde{G}} y_{kj} \log \left[\frac{1}{\tilde{G}} f_j(\mu_k | \tilde{\mu}_j, \Sigma_{\tilde{\mu}_j}) \right] + \mathcal{C}. \end{aligned} \quad (7)$$

The EM algorithm iterations to maximize this expression, and thus determine the MAP result, are as follows.

1) *E-Step*: As before, we find $\hat{z}_{ik} = E[z_{ik} | x_i, \Theta]$ given parameter estimates from the M-step,

$$\hat{z}_{ik} = \frac{\hat{\tau}_k f_k(x_i | \hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{j=1}^G \hat{\tau}_j f_j(x_i | \hat{\mu}_j, \hat{\Sigma}_j)}.$$

Now, additionally calculate the expectation of the other hidden data,

$$\hat{y}_{kj} = \frac{f_j(\mu_k | \tilde{\mu}_j, \Sigma_{\tilde{\mu}_j})}{\sum_{l=1}^{\tilde{G}} f_l(\mu_k | \tilde{\mu}_l, \Sigma_{\tilde{\mu}_l})}.$$

2) *M-Step*: As our prior term in (7) is independent of the parameters τ_k and Σ_k , these estimates remain the same as the ML version, namely ¹

$$\hat{\tau}_k = \frac{n_k}{n}$$

and

$$\hat{\Sigma}_k = \frac{1}{n_k} \sum_{i=1}^n \hat{z}_{ik} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

where $n_k = \sum_{i=1}^n \hat{z}_{ik}$. Maximizing (7) with respect to μ_k gives

$$\begin{aligned} \hat{\mu}_k &= \left[\sum_{i=1}^n \hat{z}_{ik} \hat{\Sigma}_k^{-1} + \sum_{j=1}^{\tilde{G}} \hat{y}_{kj} \Sigma_{\tilde{\mu}_j}^{-1} \right]^{-1} \\ &\quad \left[\sum_{i=1}^n \hat{z}_{ik} \hat{\Sigma}_k^{-1} x_i + \sum_{j=1}^{\tilde{G}} \hat{y}_{kj} \Sigma_{\tilde{\mu}_j}^{-1} \tilde{\mu}_j \right]. \end{aligned} \quad (8)$$

This is contrasted to the ML estimation,

$$\hat{\mu}_k = \frac{\sum_{i=1}^n \hat{z}_{ik} x_i}{\sum_{i=1}^n \hat{z}_{ik}}.$$

Note that (8) has the form of a weighted average of the data points x_i with (fuzzy) membership to cluster k and the prior means $\tilde{\mu}_j$ (fuzzily) affiliated to cluster k , with the weights governed by the respective covariance matrices. A minor drawback to the MAP parameter calculation is that (8) includes $\hat{\Sigma}_k$, implying the need to now simultaneously solve the equations for the parameters $\hat{\mu}_k$ and $\hat{\Sigma}_k$ — alternately one may use an approximation for $\hat{\Sigma}_k$ to solve (8), then find $\hat{\Sigma}_k$ in the usual way.

B. Generating Seed Clusters

Two significant issues must be considered when employing an EM algorithm for clustering. First, the algorithm requires *a priori* knowledge of the model order, or number of clusters, G — an issue shared by most clustering techniques. We adopt the following common workaround: choose a range $G = 1, \dots, G_{max}$ ² of candidate model orders, cluster for each G separately, and evaluate which worked the best.³

¹Parsimonious estimations of Σ_k are also possible [25]; an equal-volume assumption among clusters may be preferred.

²We compute G_{max} as a function of the number of data points N , with a limit of 6.

³Our evaluation is based on BIC of each candidate result; see section IV.

The second issue is that the EM method is highly susceptible to local optima near its initial values, or seed clustering. However, here we can leverage our preceding step's result to make a better initial guess of the current clusters. The basic approach to using this "prior" information in generating the seed is to assign the current data points to whichever prior cluster is closest, using the Mahalanobis distance. Recall, however, that we need a seed clustering for each of a *range* of model orders (numbers of clusters) $G = 1, \dots, G_{max}$. The primary complications arise in cases G is different from the number of prior clusters \tilde{G} .

In all cases we begin by building the n -by- \tilde{G} matrix D_M of Mahalanobis distances of each data point from each prior cluster, where the $(i, j)^{th}$ element is $d_{M,j}(x_i) = \sqrt{(x_i - \tilde{\mu}_j)^T \tilde{\Sigma}_j^{-1} (x_i - \tilde{\mu}_j)}$. Then the procedure varies for the three cases outlined below.

1) *Case $G = \tilde{G}$:* Here we simply assign each observation to the closest prior cluster: $\min_j d_{M,j}(x_i)$ for each i .

2) *Case $G < \tilde{G}$:* For this case we need to eliminate some cluster(s) compared to the last step; specifically, we have ΔG too many clusters, where $\Delta G = \tilde{G} - G$. The goal is to provide a good seed for when ΔG cluster(s) from the previous step have disappeared or perhaps have become indistinguishable in the current feature space. For example, the electrode may have drifted from a neuron whose signals we clustered in the previous step, or perhaps such a neuron ceased spiking activity.

We first assign observations to clusters as above and then evaluate a measure for contribution of each cluster j : $\psi_j = \sum_{i=1}^{n_j} p(x_i | \theta_j)$ over only the n_j observations x_i assigned to cluster j . Calculating this measure for $j = 1, \dots, \tilde{G}$, we throw out the ΔG prior clusters (columns of D_M) with the smallest ψ_j and then reassign all observations. Other measures of cluster "contribution" may be used; we choose ψ_j to favor both proximity to the prior cluster and the number of data points assigned to the cluster.

3) *Case $G > \tilde{G}$:* Here we must generate $\Delta G = G - \tilde{G}$ "extra" clusters. Clearly this is the most complicated case as it requires actually implementing a clustering step to guess the seed. The primary circumstance to consider is that ΔG new neurons have now been detected and a new cluster should be created for each. Another possibility is that the prior step's clustering result was incorrect, with multiple neurons being grouped into one cluster, and we hope this step rectifies the error.

The key idea for this case is to find one point in each of the "new" clusters by finding the worst matches to the prior clusters. Then, to generate a full seed, we implement the (straightforward and fast) k-means algorithm [26] (with $k = G$) using starting centroid locations specified as a combination of our \tilde{G} prior means and these ΔG newly identified ill-fitting points. Note these ill-fitting points must be determined sequentially — i.e. when finding the second point we seek a point far from the prior clusters *and* far from the first point — to avoid choosing two adjacent points that likely belong to the same cluster (rather than to separate

clusters). Additionally, the ΔG *worst*-fitting points likely include gross outliers. Thus, k-means runs multiple times with different starting centroid locations, with the others randomly sampled from among a set of worst-fitting points. The k-means results with the best score (sum of point-centroid distance) is used as the seed.

C. Tracking Clusters Across Steps

Ultimately our goal is to "track neurons" — that is, to associate specific neurons with specific signals over time by matching the current clusters with clusters of the preceding step (or possibly identify them as newly appearing or disappearing neurons). The clusters' statistics are used to evaluate the probability of such matches as follows, a method bearing strong resemblance to the technique in [19].

Construct the G -by- \tilde{G} matrix D_M of Mahalanobis distances between each mean μ_k and each prior cluster mean $\tilde{\mu}_j$, where the $(k, j)^{th}$ element is $d_{M,\tilde{\mu}_j}(\mu_k) = \sqrt{(\mu_k - \tilde{\mu}_j)^T \Sigma_{\tilde{\mu}_j}^{-1} (\mu_k - \tilde{\mu}_j)}$. (Other choices of metric may also be applied.) Then repeat the following until all prior clusters or all current clusters have been assigned (or until the loop is broken in step 1):

- 1) Let (k^*, j^*) denote the element with the minimum value in D_M . If this value exceeds a threshold $d_{M,max}$, break the loop; otherwise continue.
- 2) Map current cluster k^* to the neuron represented by prior cluster j^* .
- 3) Remove the k^{*th} row and j^{*th} column of D_M from consideration in succeeding steps.

Any remaining unassigned current or prior clusters are assumed to be newly appearing or disappearing neurons, respectively. The threshold value in step 1 above is designed to prevent a newly appearing neuron being mapped to a newly disappearing neuron.

IV. RESULTS

Figure 1 displays spike data from a sequence of consecutive recording steps from macaque parietal cortex, collected in an acute recording session with platinum-iridium, 1.5 M Ω -impedance electrodes in a microdrive controlled by our autonomous algorithm [2], [3], [4]. Each step contains 10 seconds of data with separating intervals of approximately 20 seconds. Spikes were detected from the recorded voltage stream according to a wavelet matching approach [27], aligned by their minimum, and projected onto a two-dimensional PCA space. We choose this particular sequence to highlight some failures of not utilizing priors in clustering.

Displayed alongside the clustering results of our algorithm are the results of the EM algorithm with ML parameters, seeded with clusters from a standard hierarchical agglomerative technique. This "baseline" algorithm follows the proposal of [23], which we have used for the past two years in hundreds of recording sessions, chosen based on its high rate of success compared to other spike sorting options. In both algorithms, we implemented a "background" mixture component of uniform distribution to capture outliers. Additionally, following the suggestion of [23], we chose the

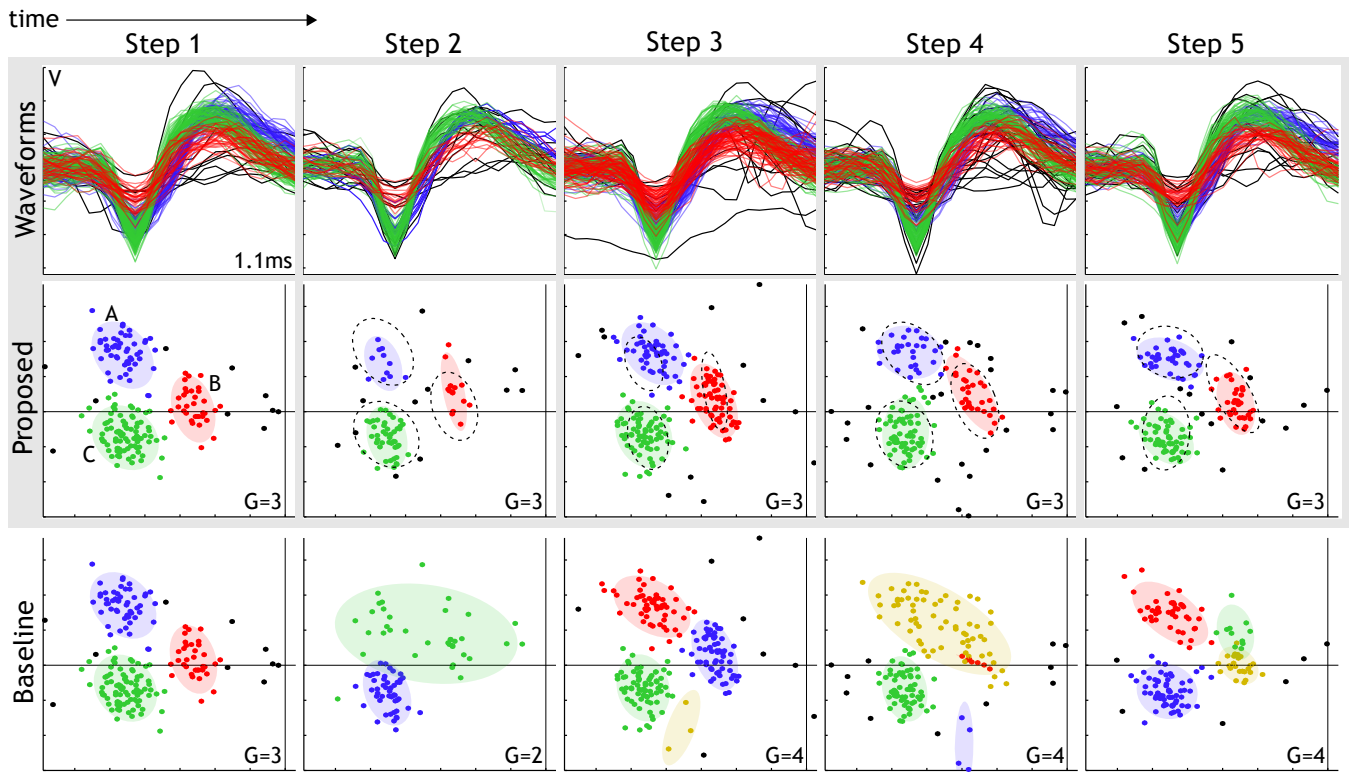


Fig. 1. Cluster results over five consecutive steps in a common PCA space. Rows: (1) Extracted, aligned waveforms from the step; (2) clustering results from our proposed algorithm; (3) clustering results from the baseline (ML) algorithm. 2-sigma ellipses of the current Gaussian model (colored, filled) and of the prior clusters (black, dashed) are superimposed on the current data. Black waveforms / PCA points have been classified as outliers.

model order giving best mixture results (over the range of candidate G) according to the Bayesian information criterion (BIC), an approximation to the Bayes factor that includes a penalty for model complexity:

$$BIC \equiv 2l_M(\hat{\Theta} | X, \mathcal{M}) - m_{\mathcal{M}} \log n,$$

for model \mathcal{M} , maximized mixture log-likelihood l_M , and number of independent model parameters $m_{\mathcal{M}}$.

In Step 1, the baseline clustering result is shown in both columns, as there is no “prior” step. Manual examination of the waveforms suggests that Step 1’s identification of three distinct neurons is accurate and that these ostensible neurons (labeled A, B, and C) persist through the remaining steps shown. The clustering challenge is difficult, however, as the spike features are not highly separated and the firing rates (and thus numbers of data points) are sometimes low. Overall, notice our proposed algorithm consistently identifies three clusters in approximately the same PCA position, whereas the baseline algorithm gives statistically sound but somewhat incongruous results, seemingly more volatile to noise variations.

In the baseline case, Steps 2 and 4 “fail” by combining the signals of neurons A and B into a single cluster. The Step 4 baseline clustering results return two extraneous clusters additionally. In Step 5, the baseline algorithm splits neuron B’s signals into two groups. Thus we see our proposed algorithm can successfully compensate for both under- and over-clustering cases. Step 3 shows a good result by both

algorithms, though some probable outliers are grouped as an independent cluster in the baseline case.

Moreover, the consistency of the cluster colors in each step signifies our algorithm’s tracking of neurons A, B, and C through the sequence of samples. No such attempt is made for the baseline case and thus the labels are essentially random.

In Figure 2, we present a detailed view of one step from a different recording session. In this case, unlike in Figure 1, the prior does not reflect the appropriate result for the current step; in the preceding step, only two clusters were identified, as the neuron represented by the left-most cluster was not present. However, our algorithm correctly clusters the data into three components, two of which match the prior locations. The baseline algorithm returns only two clusters for the current step. In this case, our seeding strategy plays a strong role in producing a more desirable result than the baseline; the baseline algorithm’s seed, as well as a randomly generated k-means seed, are shown for comparison in Figure 2.

The consistency of the clustering outcome is a primary benefit of the proposed algorithm. Although it is difficult to compellingly quantify this advantage, we chose to examine the change in the number of clusters from step to step as a rough indicator of clustering consistency. Figure 3 contains a typical plot of the number of clusters returned from the baseline and proposed algorithms over all 216 steps of an example recording session. Taking $\Phi = \sum_t |G_t - \bar{G}_t|$ over all

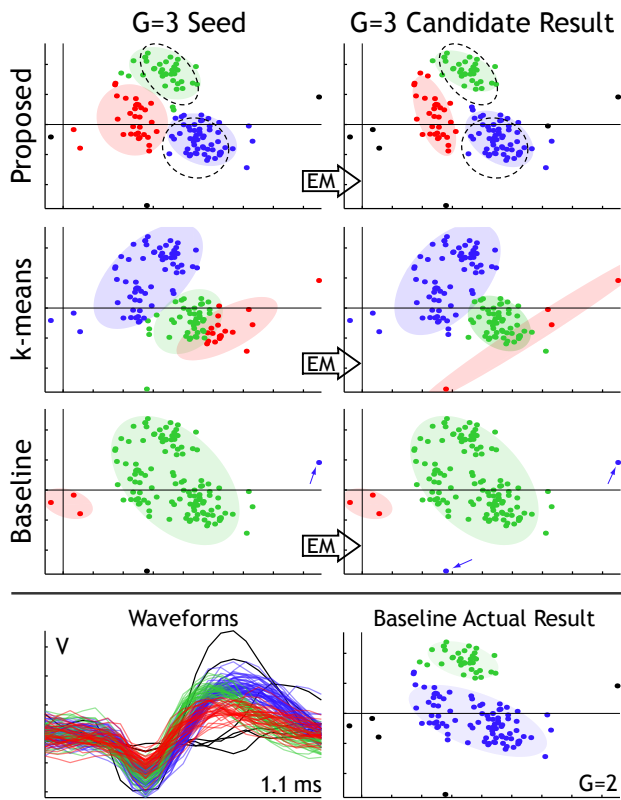


Fig. 2. Seed study of an example step with non-matching prior. The first three rows contain $G = 3$ seed clusters and resulting candidate EM output from (1) our proposed algorithm, (2) a random k-means seed, and (3) the baseline algorithm. Note the prior contained only two clusters (black dashed ellipses). Also shown are the waveforms, colored according to our algorithm's result, and the *actual* baseline result, chosen among all baseline candidates based on best BIC.

time steps t for which we applied the proposed algorithm, we get some quantitative measure of “inconsistency” (note many changes of G are correct / desired as the data change over a recording). Examining a randomly selected 100 consecutive recording sessions, comprising 10891 total time steps t , $\Phi = 4451$ for the proposed algorithm, compared to $\Phi = 7735$ for the baseline algorithm, a 42% decrease.

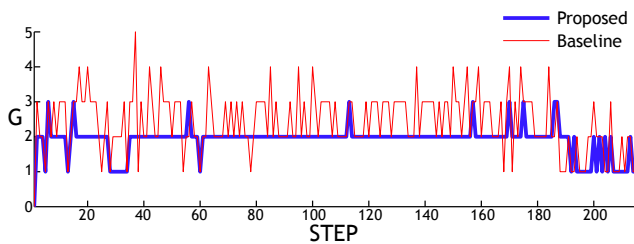


Fig. 3. Number of clusters over time for an entire recording session, comparing consistency of baseline and proposed algorithms.

V. DISCUSSION

The value of the proposed algorithm can perhaps best be understood by considering what happens when clustering is inconsistent, as in Figure 1's baseline results. Most obviously, at least to electrophysiologists, is that one can only claim a

single isolated neuron (neuron C), thus perhaps obtaining only one third of the recording's meaningful data. From the perspective of building an autonomous electrode positioning algorithm, however, a slightly more subtle repercussion exists: Suppose we are attempting to autonomously maximize the SNR of neuron A. Then, in steps where clustering of neuron A is inaccurate (2 and 4), the resulting SNRs are artificially altered — reduced by averaging the SNR of both neurons A and B — and the algorithm's deduction of the ideal electrode position will be false. Finally, if decoding these neurons' spike rates for controlling a prosthesis, the spike rate data can be similarly corrupted due to poor clustering.

Recall that the final output of each step's clustering is not just the EM optimization of the mixture model, but also the determination of which candidate model (for $G = 1, \dots, G_{max}$) is best. Two common error modes exist in other EM algorithms for choosing the wrong model order. Suppose the appropriate model order is $G = 3$. The first error mode occurs when the EM result for $G = 3$ is caught in a poor local optimum, and thus its BIC score is worse than other results. Figure 2 shows an example of this type of result, with the hierarchical seed making an outlier its own seed cluster. Secondly, we have observed cases where, although the result for, say, $G = 3$ matched the desired result, the BIC score for another mathematically reasonable clustering (say, $G = 4$) was slightly better. With our algorithm, not only are we more likely to avoid local optima because of the seeding technique, but also the BIC score itself is moderately influenced by the prior clusters via the MAP optimization.

Our algorithm has also performed well where the prior is not similar to the current clusters. Neurons appearing on one time step are not guaranteed to fire during the next sampling interval, and applied clustering and tracking algorithms must account for the addition or reduction of clusters (or both), as discussed for Figure 2. The prior's construction as a mixture of densities effectively influences the cluster locations but assumes neither a certain number of clusters nor the association of *particular* current and prior clusters. Thus, the algorithm is not unduly biased by the prior when evidence suggests the appearance (or disappearance) or neurons.

Finally, we note the need for maintaining fast computation speed, particularly as we aim for real-time autonomous electrode positioning. Although we have introduced complexity versus the baseline ML method, our seed clusters tend to be much closer to the (at least local) optimum, and thus the EM algorithm requires usually very few iterations to converge. In fact, average computation time decreased nearly 50% from the baseline.

A few important elements are left as future work. First, we assume we have a preceding step with “reasonable” clustering results. We could consider in more depth how these results are generated, perhaps sacrificing computation time to ensure we start with a good prior. Additionally, we may not wish to consider all neurons “equal” as we have in this paper. For example, if one neuron's SNR is higher and we

wish to track that particular neuron, perhaps we match it first in the tracking procedure and are not as likely to drop it for the $G < \tilde{G}$ case seeding. Finally, some clustering mistakes (or temporarily silent neurons) are inevitable. We look to make the tracking algorithm more robust by incorporating prior information from several time steps and implementing a multiple hypothesis tracking approach.

VI. ACKNOWLEDGEMENTS

We thank Richard Andersen and the members of his lab, particularly Grant Mulliken for collaboration and test data. This work is funded by the National Institutes of Health, grant R01 EY015545.

REFERENCES

- [1] M. Linderman, B. Gilja, G. Santhanam, A. Afshar, S. Ryu, T. Meng, and K. Shenoy, "Neural recording stability of chronic electrode arrays in freely behaving primates. Program No. 13.7," in *Abstract Viewer / Itinerary Planner*. Atlanta, GA: Society for Neuroscience, 2006.
- [2] J. G. Cham, E. A. Branchaud, Z. Nenadic, B. Greger, R. A. Andersen, and J. W. Burdick, "Semi-chronic motorized microdrive and control algorithm for autonomously isolating and maintaining optimal extracellular action potentials," *Journal of Neurophysiology*, vol. 93, pp. 570–579, January 2005.
- [3] Z. Nenadic and J. W. Burdick, "A control algorithm for autonomous optimization of extracellular recordings," *IEEE Transactions on Biomedical Engineering*, vol. 53, no. 5, pp. 941–955, May 2006.
- [4] E. A. Branchaud, "An algorithm for the autonomous isolation of neurons in extracellular recordings," Ph.D. dissertation, California Institute of Technology, 2006.
- [5] J. G. Cham, M. T. Wolf, R. A. Andersen, and J. W. Burdick, "Miniature neural interface microdrive using parylene-coated layered manufacturing," in *Proc. IEEE/RAS-EMBS Intl. Conf. on Biomedical Robotics and Biomechanics*, 2006.
- [6] M. S. Lewicki, "A review of methods for spike sorting: the detection and classification of neural action potentials," *Network: Computation in Neural Systems*, vol. 9, pp. R53–R78, 1998.
- [7] M. S. Fee, P. P. Mitra, and D. Kleinfeld, "Automatic sorting of multiple unit neuronal signals in the presence of anisotropic and non-Gaussian variability," *Journal of Neuroscience Methods*, vol. 69, no. 2, pp. 175–188, Nov. 1996.
- [8] M. Salganicoff, M. Sarna, L. Sax, and G. Gerstein, "Unsupervised waveform classification for multi-neuron recordings: a real-time, software-based system. i. algorithms and implementation," *Journal of Neuroscience Methods*, vol. 25, no. 3, pp. 181–187, Oct. 1988.
- [9] F. Ohberg, H. Johansson, M. Bergenheim, J. Pedersen, and M. Djupsjöbacka, "A neural network approach to real-time spike discrimination during simultaneous recording from several multi-unit nerve filaments," *Journal of Neuroscience Methods*, vol. 64, no. 2, pp. 181–187, Feb. 1996.
- [10] A. Bar-Hillel, A. Spiro, and E. Stark, "Spike sorting: Bayesian clustering of non-stationary data," *Journal of Neuroscience Methods*, vol. 157, no. 2, pp. 303–316, Oct. 2006.
- [11] S. Shoham, M. R. Fellows, and R. A. Normann, "Robust, automatic spike sorting using mixtures of multivariate t-distributions," *Journal of Neuroscience Methods*, vol. 127, no. 2, pp. 111–122, Aug. 2003.
- [12] J. C. Letelier and P. P. Weber, "Spike sorting based on discrete wavelet transform coefficients," *Journal of Neuroscience Methods*, vol. 101, no. 2, pp. 93–106, Sept. 2000.
- [13] E. Hulata, R. Segev, and E. Ben-Jacob, "A method for spike sorting and detection based on wavelet packets and Shannon's mutual information," *Journal of Neuroscience Methods*, vol. 117, no. 1, pp. 1–12, May 2002.
- [14] R. Q. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.
- [15] A. Pavlov, V. A. Makarov, I. Makarova, and F. Panetos, "Separation of extracellular spikes: When wavelet based methods outperform the principle component analysis," *Lecture Notes in Computer Science*, vol. 3561, pp. 123–132, Oct. 2005.
- [16] T. I. Aksenova, O. K. Chibirova, O. A. Dryga, I. V. Tetko, A.-L. Benabid, and A. E. P. Villa, "An unsupervised automatic method for sorting neuronal spike waveforms in awake and freely moving animals," *Methods*, vol. 30, no. 2, pp. 178–187, June 2003.
- [17] M. S. Lewicki, "Bayesian modeling and classification of neural signals," *Neural Computation*, vol. 6, pp. 1005–1030, 1994.
- [18] R. K. Snider and A. B. Bonds, "Classification of non-stationary neural signals," *Journal of Neuroscience Methods*, vol. 84, no. 1-2, pp. 155–166, Oct. 1998.
- [19] A. Emondi, S. Rebrik, A. Kurgansky, and K. Miller, "Tracking neurons recorded from tetrodes across time," *Journal of Neuroscience Methods*, vol. 135, pp. 95–105, 2004.
- [20] G. McLachlan and D. Peel, *Finite Mixture Models*. Wiley Interscience, 2000.
- [21] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.
- [22] P. Cheeseman and J. Stutz, "Bayesian classification (AutoClass): Theory and results," in *Advances in Knowledge Discovery and Data Mining*, U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy, Eds. Cambridge, MA: AAAI/MIT Press, 1996, ch. 6, pp. 61–83.
- [23] C. Fraley and A. E. Raftery, "How many clusters? Which clustering method? Answers via model-based cluster analysis," *Computer Journal*, 1998.
- [24] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.
- [25] G. Celeux and G. Govaert, "Gaussian parsimonious clustering models," *Pattern Recognition*, vol. 28, pp. 781–793, 1995.
- [26] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. Wiley Interscience, 2000.
- [27] Z. Nenadic and J. W. Burdick, "Spike detection using the continuous wavelet transform," *IEEE Transactions on Biomedical Engineering*, vol. 52, no. 1, pp. 74–87, January 2005.